Advances in Data Mining: Assignment 2

Group 4

October 8, 2024

1 Introduction

In 2018, a Kaggle competition was made, called 'Predict Future Sales'. In this competition the main task is to predict total sales for each product for the next month. In Sec. 2 we describe the given data and we visualise it in Sec. 3. Sec. 4 presents how we predicted the sales. The conclusion of this project can be found in Sec. 5.

2 Problem Description

Our data consists of 6 csv files. The first one is "sales_train.csv" which contains the daily historical data of from January 2013 to October 2015. For each sale we have information about the "date" that the sale was made as well as the "date_block_num" which groups all the sales made within the same month, starting from 0 for the first month and going all the way to 33 for the last month. There is also the "shop_id" for the shop that made the sale, the "item_id" indicating which item was sold and "item_price" for the price of the item sold. The last row "item_cnt_day" tells us how many items were sold, with negative numbers indicating that the shop bought that many items.

The "test.csv" contains the data we need to make predictions for the month of November 2015. It has the "shop_id" and the "item_id" for each possible combination, which is also uniquely identified by the "ID" column. Our job is to predict the "item_cnt_month" for each one, meaning the number of items sold within the month of November 2015. This is also clear by looking at the "sample_submission.csv" which gives us the correct format of the final submission containing two columns, the "ID" which matches the one in the testing set and the "item_cnt_month" which is our prediction.

The "items.csv" file contains more information about each product that is being sold. For each one we have the "item_name" with the full name of the product, the "item_id" with the id for the specific product and the "item_category_id" giving us the category that it belongs to. The "item_categories.csv" gives us the description of each product category. In total we have 84 different categories and for each one we have the "item_category_id" and the "item_category_name", which is the full name of each product category. Finally, "shops.csv" contains information about each individual shop. The two columns gives us the "shop_name", meaning the full name of each shop and the "shop_id", the id by which we will identify each shop.

Our objective is to generate predictions for the month of November 2015. The predictions will be the number of expected sales for each combination of shop and item. As we already described each unique combination of this form is characterized by a unique ID, which we can find in "test.csv".

3 Visualisation

In Fig. 1 we present the item sales each year. The year of 2013 and 2014 show an overall increasing trend, whereas in 2015 the item sales dropped at the end of the year. The number of items in each numbered category is shown in Fig. 2. As we can see the peaks are at item 0 and 35. The basic statistics of the item prices be read in Table 1. The mean and the median are relatively low (¡ 1000), however the variance is extremely large, as there are several items with unrealistic values. These prices are plotted against the ID of the item in Fig. 3. The plot is a zoomed in version of the original plot, as there is one item with a price of 307980. The basic statistics of the number of items sold each day is presented in Table 2, while we plot the daily sales of each item in Fig. 4. In this plot, we again decided to zoom in, as there is one value in the item_cnt_day list, which is 2169. As the mean (1.0), median (1.24), and variance (6.85) show (see Tab. 2), this value is extremely large compared to the other values. Since our task is to determine the future sales for the next month, we present the monthly sells of each item ID in Fig. 5 with a minimum and maximum item_cnt_month of 0 and 20, respectively. The reason behind that is that

99.8% of the values fall below 20. Furthermore, we show the statistics of the item_cnt_month before and after putting these values in this certain threshold (i.e. clipping) in Tab. 3 and Tab. 4, respectively. As expected, before clipping, the mean and the variance are larger. However, the median does not change as it is 0 at both cases.

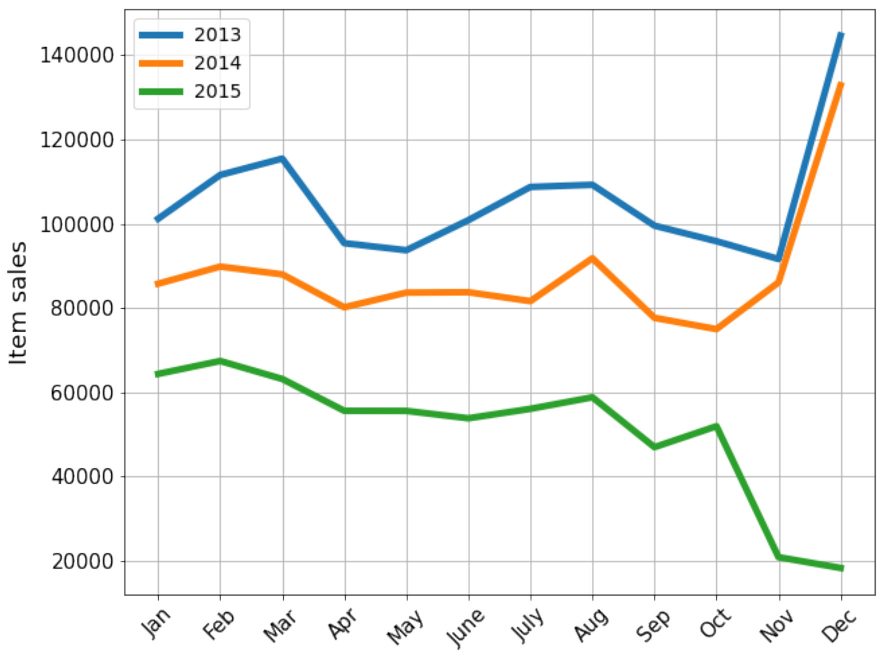

Figure 1: Item sales as a function of months. The different colours indicate different years as labeled.

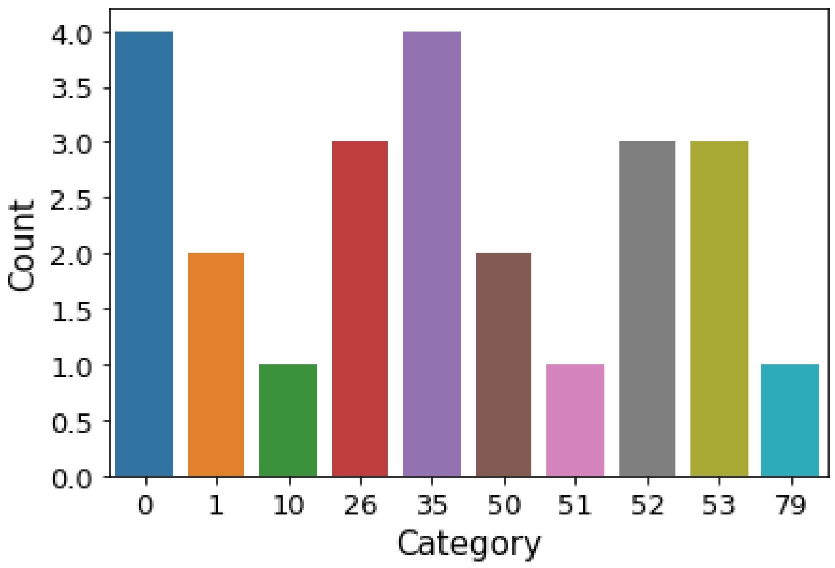

Figure 2: Item count per category.

Item price	Mean	Median	Variance
	399	890	2992205

Table 1: The median, mean, and variance of the item prices.

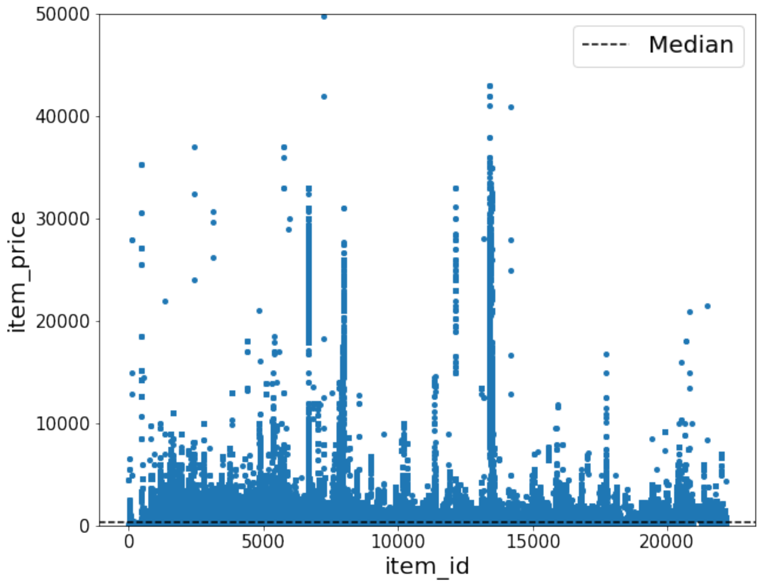

Figure 3: Item prices as a function of their ID. The median of the prices is shown with a dashed line.

Item count day	Mean	Median	Variance
	1.0	1.24	6.85

Table 2: The mean, median, and variance of the daily sales.

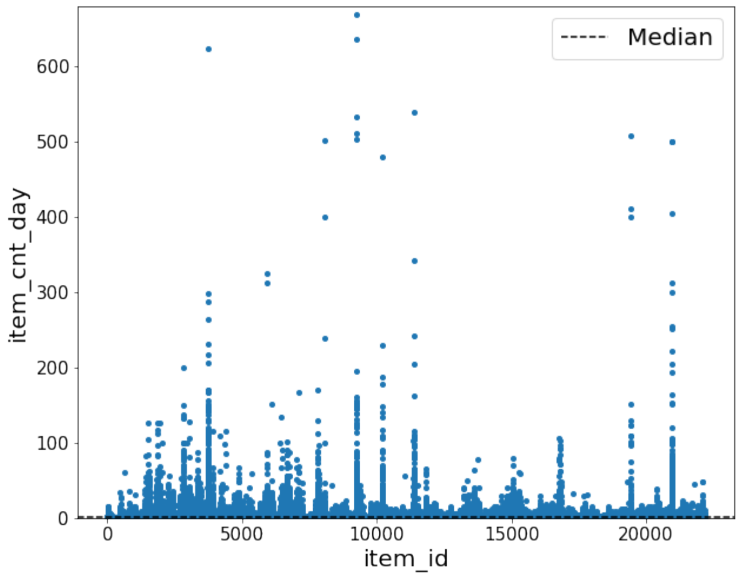

Figure 4: The daily sales versus the ID of the item. Black dashed line indicates the median of the daily sales.

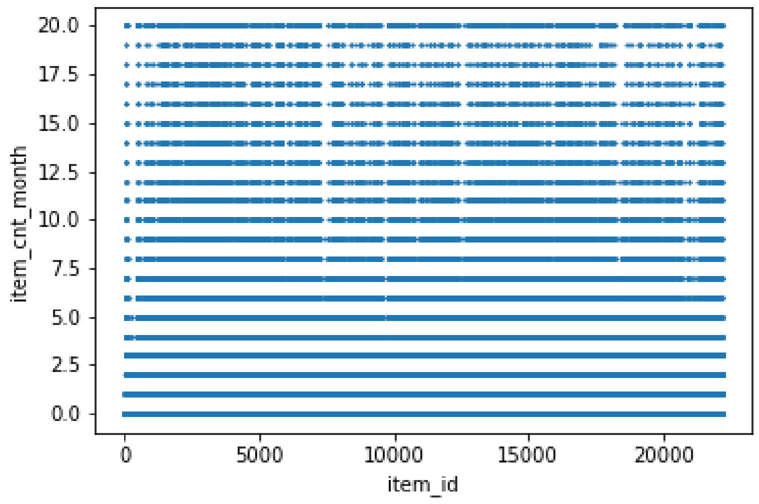

Figure 5: The number of items sold each month. The values are put in a threshold of 0 and 20 (see text for details).

Item count month	Mean	Median	Variance
	0.32	0.0	11.45

Table 3: The mean, median, and variance before clipping.

Item count month	Mean	Median	Variance
	0.29	0.0	1.46

Table 4: The mean, median, and variance after clipping.

4 Experimental setup

Before beginning to apply the forecasting algorithms we need to create the training, validation and testing data sets. Because we don't have access to the correct values of the predictions that we will generate, we decided to use the last available month (October 2015) as our validation set. The purpose of this set is that we will exclude it from the training process and only use it as an indicator of what our final score should be.

Our training set contains all the entries that correspond to months ranging from January 2013 to September 2015, which is consistent with date_block_num from 0 to 32. The X_train is multidimensional and we can include many different parameters that we think will be a valuable indicator of future sales of the product. We chose to use the date_block_num , shop_id and item_id . We also used an extended version which included the item_category_id, item_maincategory_id and item_subcategory_id. The last two IDs were created based on the name of the item category that had two names separated with a "-". We used the LabelEncoder to generate integers that will act as a unique identifier of each of the two names of the item's category.

The Y_train includes the values of "item_cnt_month" that correspond to each input of X_train. The value of "item_cnt_month" tells us the total number of sales for the corresponding compination of item and shop for each month. It is clipped between 0 and 20 as discussed in Section 3 to remove the outliers which will give us better results in our predictions.

4.1 LGBM Regressor

LightGBM is an open-source gradient boosting framework that is based on the tree learning technique and is aimed to handle data more quickly and accurately. It can handle massive datasets while using less memory and allows distributed learning.

After transforming our training and validation sets to a form that can be used to train the Regressor we train for 50 iterations with an early stopping of 5 rounds, meaning if the score on the validation set doesn't improve for 5 consecutive training rounds, the process will be terminated and the best scoring iteration will be used.

The results, including the training time, can be found in Table 5. The score for the training and validations set was calculated during the training process. The score for the testing set which is our actual goal could only be calculated after generating the predictions using the X_test and submitting it on the Kaggle challenge.

	Training	Validation	Testing	Training Time
Score	1.39	1.24	1.19	20.56 s

Table 5: The final scores for the training, validation and testing sets using the LGBM Regressor. The last column shows the time required for the training process.

We repeat the same process using the extended dataset that contains more dimensions for the input of the model. The results are shown in Table 6

	Training	Validation	Testing	Training Time
Score	1.13	1.02	1.098	24.39 s

Table 6: The final scores for the extended versions of the training, validation and testing sets using the LGBM Regressor. The last column shows the time required for the training process.

We see that increasing the dimensions of the input to include even more information led to a noticeable improvement on the final testing score.

4.2 XGBoost Algorithm

Xgboost (eXtreme Gradient Boosting) is an improved version of the ensemble machine learning method that uses the gradient descent boosting approach. It is applicable to both regression and classification issues. Its purpose is to maximize model performance as well as execution speed. It is ten times quicker than standard Gradient Boosting. Furthermore, XGBoost incorporates a novel split-finding technique to optimize trees, as well as built-in regularization to minimize over fitting. For our implementation we use the XGBRegressor function.

We train using 500 estimators and an early stopping of 10 rounds. After the training process is over we use the X_test data set in order to generate the predictions we want to submit on the Kaggle challenge. The results for all the data sets can be found in Table 7. It also includes the time required to train the Regressor.

	Training	Validation	Testing	Training Time
Score	1.18	1.11	1.18	705.53 s

Table 7: The final scores for the training, validation and testing sets using the XGBoost Algorithm. The last column shows the time required for the training process.

We repeat the same process using the extended data sets. The results are shown in Table 8

	Training	Validation	Testing	Training Time
Score	1.07	1.00	1.080	853.44 s

Table 8: The final scores for the extended versions of the training, validation and testing sets using the XGBoost Algorithm. The last column shows the time required for the training process.

Similarly to the previous method we also had a noticeable improvement of our final testing score with the extended data set. This however came at the cost of some more computational time that was required during the training process.

One thing that will help us visualise the training process is the importance of each of the training features. In Figure 6 we can find the importance of each individual dimension of our multidimensional input during the training of our algorithm. The three features that were included in the extended version of our data sets but are not part of the original ones are dominating the other three. However this can lead to some over-fitting and this is the reason that including more and more features will not yield better results.

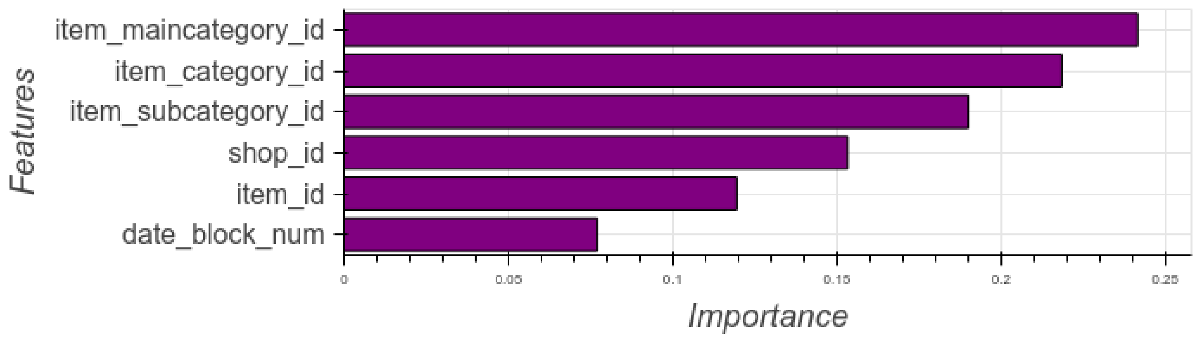

Figure 6: The importance of each feature that was used for the training of the XGBoost Algorithm. This graph was created for the extended version of our training input.

4.3 Random Forest Regression Algorithm

Random Forest Regression is a supervised learning technique that uses ensemble learning to do regression by combining predictions from multiple machine learning algorithms to generate a more accurate prediction than a single model. During training, a Random Forest algorithm constructs many decision trees and outputs the mean of the classes as the forecast of all the trees.

We trained our random forest regression model using the sklearn package, specifically the RandomForestRegressor function. Our base model uses 40 trees, the max depth of each tree is set to 15 and the random_state is set to 42 in order to make the code reproducible.

We train the model using our training set. After the training process is over we calculate the scores for both the training and the validation sets. The final step is to use our model and the testing set in order to generate predictions that are ready to be submitted on Kaggle challenge. The final results can be found in Table 9.

	Training	Validation	Testing	Training Time
Score	0.24	0.15	1.119	678.73 s

Table 9: The final scores for the training, validation and testing sets using the Random Forest Regression Algorithm. The last column shows the time required for the training process.

We repeat the same process but this time we use the extended versions of our data sets. The results can be found in Table 10.

	Training	Validation	Testing	Training Time
Score	0.41	0.21	1.099	945.32 s

Table 10: The final scores for the extended versions of the training, validation and testing sets using the Random Forest Regression Algorithm. The last column shows the time required for the training process.

As we can see there is once again a slight improvement on the score of the testing set. This however came at the cost of more computational time.

5 Conclusions

During this assignment we managed to succesfully complete the Kaggle challenge “Predict Future Sales”. Our best score is 1.080 which was achieved by implementing XGBoost Algorithm for our extended data set. We also created various visualisation plots in order to better understand the data we were given.

As we already discussed increasing the number of dimensions of our input to include more training features can lead to better results. There is however the issue of over-fitting the data. When we attempted to increase the number of features even further, the final scores on the testing set didn’t improve and in most cases they got even worse than our original scores before the extended version.

In Figure 7 we can see the final results for all the three different forecasting algorithms that we used. The results are for the extended data sets that gave us the best results overall.

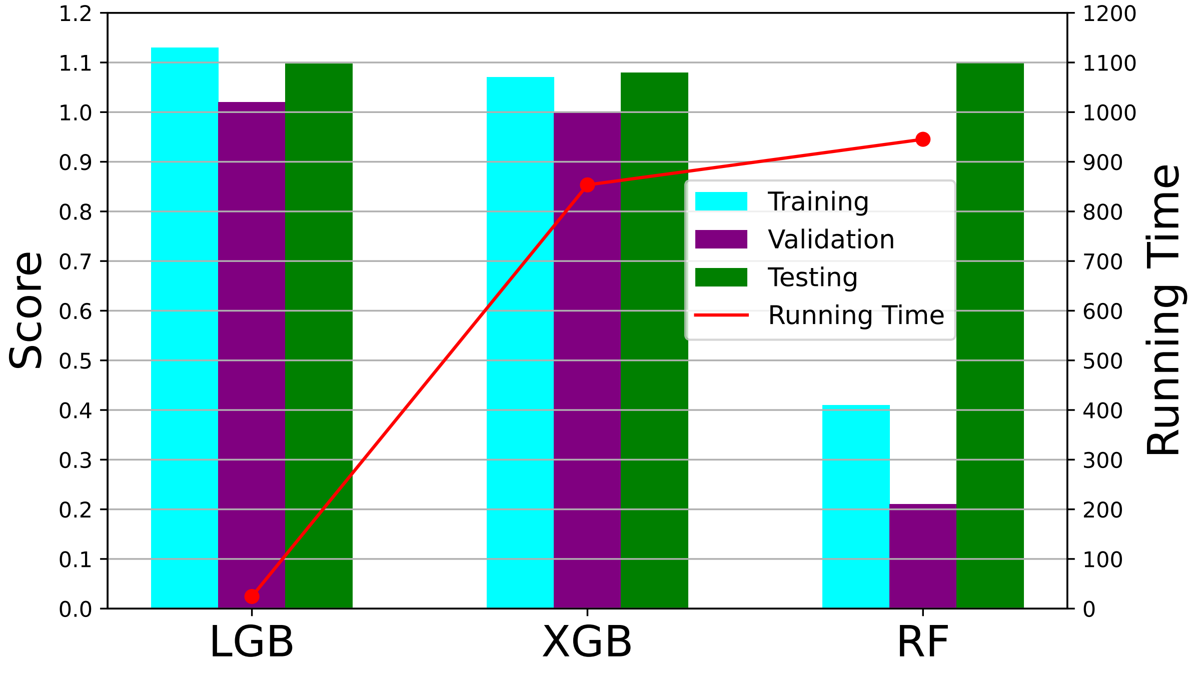

Figure 7: The final results for all the tree different algorithms that were implemented. The results include the score of the training and validation set as well as the score of the testing set as it was calculated on Kaggle. The red line shows the running time for the training process of each algorithm.

On the Kaggle competition our team name is "**group_4_leiden_aidm**".